



```

LL               IIIIII   SSSSSSSS
LL              II       SSSSSSSS
LL             I         SS
LL            I          SS
LL           I           SS
LL          I            SS
LL         I             SS
LL        I              SS
LL       I               SS
LL      I                SS
LL     I                 SS
LL    I                  SS
LL   I                   SS
LLL LLLLLLLLLL   IIIII   SSSSSSSS
LLLLLLLLLLLLL   IIIII   SSSSSSSS
```

```
1 0001 0 MODULE show$network (IDENT = 'V04-000') =
2 0002 1 BEGIN
3 0003 1
4 0004 1
5 0005 1
6 0006 1
7 0007 1
8 0008 1
9 0009 1
10 0010 1
11 0011 1
12 0012 1
13 0013 1
14 0014 1
15 0015 1
16 0016 1
17 0017 1
18 0018 1
19 0019 1
20 0020 1
21 0021 1
22 0022 1
23 0023 1
24 0024 1
25 0025 1
26 0026 1
27 0027 1
28 0028 1
29 0029 1
30 0030 1
31 0031 1
32 0032 1
33 0033 1
34 0034 1
35 0035 1
36 0036 1
37 0037 1
38 0038 1
39 0039 1
40 0040 1
41 0041 1
42 0042 1
43 0043 1
44 0044 1
45 0045 1
46 0046 1
47 0047 1
48 0048 1
49 0049 1
50 0050 1
51 0051 1
52 0052 1
53 0053 1
54 0054 1
55 0055 1
56 0056 1
57 0057 1

*****
*
*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*  ALL RIGHTS RESERVED.
*
*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
*  TRANSFERRED.
*
*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
*  CORPORATION.
*
*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

++
FACILITY:  SHOW Command

ABSTRACT:

    This module processes the SHOW NETWORK command

ENVIRONMENT:

    VAX/VMS operating system. unprivileged user mode,

AUTHOR:  Tim Halvorsen, August 1981

Modified by:

    V03-010 TMH0010      Tim Halvorsen   27-Jun-1983
                   Make endnode display look better.

    V03-009 TMH0009      Tim Halvorsen   17-May-1983
                   Fix bug in routine which obtains the next node name
                   in the area display.  It was accidentally sending
                   a binary count to the terminal.

    V03-008 TMH0008      Tim Halvorsen   13-Mar-1983
                   Do not display loop nodes, and add new area display.

    V03-007 GAS0105      Gerry Smith     20-Jan-1983
                   Fix output display.

    V03-006 GAS0100      Gerry Smith     11-Jan-1983
```



```

58      0058 1  Remove reference to SHOW$L_STATUS, since all
59      0059 1  errors are signaled.
60      0060 1
61      0061 1  V03-005 GAS0099      Gerry Smith      7-Jan-1983
62      0062 1  Minor modifications to fit new SHOW image.
63      0063 1
64      0064 1  V004      MKP0001      Kathy Perko      14-Dec-1982
65      0065 1  Add capability to get multiple nodes in one QIO to NETACP.
66      0066 1
67      0067 1  V003      TMH0003      Tim Halvorsen    28-Nov-1982
68      0068 1  Add formatting of area node addresses.
69      0069 1
70      0070 1  V002      TMH0002      Tim Halvorsen    24-Jun-1982
71      0071 1  Fix failure to initialize an NFB field.
72      0072 1
73      0073 1  V001      TMH0001      Tim Halvorsen    03-Jun-1982
74      0074 1  Modify to use new NETACP control QIO interface.
75      0075 1  --
76      0076 1
77      0077 1  Include files
78      0078 1
79      0079 1
80      0080 1
81      0081 1  LIBRARY 'SYSS$LIBRARY:STARLET';      ! VAX/VMS common definitions
82      0082 1
83      0083 1  LIBRARY 'SHRLIB$:NET';              ! NETACP control QIO definitions
84      0084 1
85      0085 1  REQUIRE 'SYSS$LIBRARY:UTILDEF';      ! Common BLISS definitions

```

```

87      0261 1 |
88      0262 1 | Table of contents
89      0263 1 |
90      0264 1 |
91      0265 1 FORWARD ROUTINE
92      0266 1   show$network: NOVALUE, | Process SHOW NETWORK
93      0267 1   display_nodes: NOVALUE, | Produce reachable node display
94      0268 1   format_area_info, | Write area info to the display
95      0269 1   format_node_info, | Write node info to the display
96      0270 1   get_node_name, | Get node name given node address
97      0271 1   write_line: NOVALUE, | Write line to output
98      0272 1   format_nodeadr; | Format a node address
99      0273 1 |
100     0274 1 |
101     0275 1 | OWN storage
102     0276 1 |
103     0277 1 |
104     0278 1 OWN
105     0279 1   channel: WORD; | Channel to ACP
106     0280 1 |
107     0281 1 |
108     0282 1 | Status codes
109     0283 1 |
110     0284 1 |
111     0285 1 EXTERNAL LITERAL
112     0286 1   show$_nonet; | Network not available
113     0287 1 |
114     0288 1 |
115     0289 1 | External routine
116     0290 1 |
117     0291 1 |
118     0292 1 EXTERNAL ROUTINE
119     0293 1   show$write_line: NOVALUE; | General SHOW FAO output routine

```

```
121 0294 1 GLOBAL ROUTINE show$network : NOVALUE =
122 0295 1
123 0296 1 ---
124 0297 1
125 0298 1 This routine processes the SHOW NETWORK command
126 0299 1
127 0300 1 Inputs:
128 0301 1
129 0302 1 None
130 0303 1
131 0304 1 Outputs:
132 0305 1
133 0306 1 None
134 0307 1 ---
135 0308 1
136 0309 2 BEGIN
137 0310 2
138 0311 2 LITERAL
139 0312 2 buffer_size = 512; ! Size of return buffer.
140 0313 2
141 0314 2 LOCAL
142 0315 2 nfb: BBLOCK [nfb$c_length+20*4], ! Network function block
143 0316 2 ! (room for 20 field requests)
144 0317 2 nfb_desc: VECTOR [2], ! Descriptor of NFB
145 0318 2 iosb: BBLOCK [8], ! I/O status block
146 0319 2 total_count, ! Number of entries displayed
147 0320 2 buffer_count, ! Number of entries returned in buffer
148 0321 2 buffer: BBLOCK [buffer_size], ! Return buffer
149 0322 2 buffer_desc: VECTOR [2], ! Descriptor of above buffer
150 0323 2 buffer_ptr, ! Pointer to return buffer
151 0324 2 keys: BBLOCK [4+8+nfb$c_ctx_size], ! Buffer for search keys & context
152 0325 2 key_desc: VECTOR [2], ! Descriptor of above buffer
153 0326 2 node_name_buffer: VECTOR [32,BYTE], ! Node name buffer
154 0327 2 node_name: VECTOR [2], ! Descriptor of above buffer
155 0328 2 exec_type, ! Executor node type
156 0329 2 exec_addr, ! Executor address
157 0330 2 exec_name_buffer: VECTOR [32,BYTE], ! Executor name buffer
158 0331 2 exec_name: VECTOR [2], ! Executor node name descriptor
159 0332 2 status;
160 0333 2
161 0334 2 !
162 0335 2 ! Assign a channel to the network ACP
163 0336 2 !
164 0337 2
165 P 0338 2 status = $ASSIGN(CHAN=channel, ! Assign channel to NETACP
166 0339 2 DEVNAM=XASCII('_NET:'));
167 0340 2
168 0341 2 IF NOT .status ! If error detected,
169 0342 2 THEN
170 0343 2 BEGIN
171 0344 2 IF .status EQL ss$_nosuchdev ! If network not yet up,
172 0345 2 THEN SIGNAL(show$_nonet) ! then tell user
173 0346 2 ELSE SIGNAL(.status); ! Else, report the status
174 0347 2 RETURN;
175 0348 2 END;
176 0349 2
177 0350 2 !
```



```
178 0351 2 ! Get our executor node name, address and type
179 0352 2 !
180 0353 2
181 0354 key_desc [0] = 4 + nfb$sc_ctx_size; ! Longword overhead, NO search values
182 0355 key_desc [1] = keys; ! and fixed context area
183 0356
184 0357 keys [0,0,32,0] = 0; ! Zero count of fields in P4 (unused)
185 0358 keys [4,0,16,0] = 0; ! Start key = at beginning
186 0359
187 0360 buffer_desc [0] = buffer_size; ! Setup descriptor of P4 buffer
188 0361 buffer_desc [1] = buffer;
189 0362
190 0363 CH$FILL(0,nfb$sc_length,nfb); ! Pre-zero NFB fields
191 0364
192 0365 nfb [nfb$b_fct] = nfb$sc_fc_show; ! Request 'show' function
193 0366 nfb [nfb$b_database] = nfb$sc_db_lni; ! of executor database
194 0367
195 0368 nfb_desc [0] = $BYTEOFFSET(nfb$_fldid) + 3*4; ! Construct descriptor of NFB
196 0369 nfb_desc [1] = nfb;
197 0370
198 0371 CH$MOVE(3*4, UPLIT LONG( ! Request the following fields:
199 0372 nfb$sc_lni_add, ! Executor address
200 0373 nfb$sc_lni_ety, ! Executor type
201 0374 nfb$sc_lni_nam, ! Executor name
202 0375 nfb [nfb$_fldid]);
203 0376
204 P 0377 status = $QIOW(FUNC = IOS$ACPCONTROL, ! Issue control function
205 P 0378 CHAN = .channel,
206 P 0379 IOSB = iosb,
207 P 0380 P1 = nfb_desc, ! Address of NDB descriptor
208 0381 P2 = key_desc, ! Address of key buffer descriptor
209 0382 P4 = buffer_desc); ! Address of return buffer descriptor
210 0383
211 0384 IF NOT .status ! If error detected,
212 0385 OR NOT (status = .iosb [0,0,16,0])
213 0386 THEN
214 0387 BEGIN
215 0388 IF .status EQL ss$devnotmount ! If ACP not yet started,
216 0389 THEN SIGNAL(show$_nonet) ! then indicate network not up
217 0390 ELSE SIGNAL(.status); ! Else, report the status
218 0391 RETURN;
219 0392 END;
220 0393
221 0394 exec_addr = .buffer [0,0,32,0]; ! Save our node address
222 0395 exec_type = .buffer [4,0,32,0]; ! Save our node type
223 0396 exec_name [0] = .buffer [8,0,16,0]; ! Construct descriptor of executor name
224 0397 exec_name [1] = exec_name_buffer;
225 0398 CH$MOVE(.exec_name [0], buffer+10, .exec_name [1]);
226 0399
227 0400
228 0401 ! Display title lines
229 0402
230 0403
231 0404 write_line(%ASCII 'VAX/VMS Network status for local node !AS !AS on !%D',
232 0405 format_nodeadr(.exec_addr),
233 0406 exec_name,
234 0407 0);
```

```
235 0408 write_line(%ASCID '');
236 0409
237 0410
238 0411 If we are a level 2 (area) router, then display cost/hops information
239 0412 for all areas in the network.
240 0413
241 0414 If we are a level 1 router, then the area database will display the
242 0415 "nearest level 2 router".
243 0416
244 0417
245 0418 buffer_desc [0] = buffer_size; ! Construct descriptor of return buffer
246 0419 buffer_desc [1] = buffer;
247 0420
248 0421 key_desc [0] = 4 + 4 + nfb$sc_ctx_size; ! Longword overhead, ONE search value
249 0422 key_desc [1] = keys; ! and fixed context area
250 0423
251 0424 keys [0,0,32,0] = 0; ! Zero count of fields in P4 (unused)
252 0425 keys [4,0,32,0] = true; ! REA search value EQL TRUE
253 0426 keys [8,0,16,0] = 0; ! Start key = at beginning
254 0427
255 0428 CH$FILL(0,nfb$sc_length,nfb); ! Pre-zero NFB fields
256 0429
257 0430 nfb [nfb$b_fct] = nfb$sc_fc_show; ! Request "show" function
258 0431 nfb [nfb$b_database] = nfb$sc_db_ari; ! of area database
259 0432 nfb [nfb$b_flags] = nfb$sc_mult; ! Request multiple entries per QIO
260 0433 nfb [nfb$b_srch_key] = nfb$sc_ari_rea; ! Only return reachable areas
261 0434 nfb [nfb$b_oper] = nfb$sc_op_eql; ! by checking if field EQL P2 value
262 0435
263 0436 nfb_desc [0] = $BYTEOFFSET(nfb$b_fldid) + 5*4; ! Construct descriptor of NFB
264 0437 nfb_desc [1] = nfb;
265 0438
266 0439 CH$MOVE(5*4, UPLIT LONG( ! Request the following fields:
267 0440 nfb$sc_ari_add, ! Area number
268 0441 nfb$sc_ari_dco, ! Destination cost
269 0442 nfb$sc_ari_dho, ! Destination hops
270 0443 nfb$sc_ari_nnd, ! Next node to destination
271 0444 nfb$sc_ari_dli, ! Destination circuit name
272 0445 nfb [nfb$b_fldid]));
273 0446
274 0447 total_count = 0; ! Initialize area count
275 0448
276 0449 WHILE true
277 0450 DO
278 0451 BEGIN
279 0452 status = $QIOW(FUNC = IOS$ACPCONTROL, ! Issue control function
280 0453 CHAN = .channel,
281 0454 IOSB = iosb,
282 0455 P1 = nfb_desc, ! Address of NDB descriptor
283 0456 P2 = key_desc, ! Address of key buffer descriptor
284 0457 P4 = buffer_desc); ! Address of return buffer descriptor
285 0458
286 0459 IF NOT .status ! If error detected,
287 0460 OR NOT (status = .iosb [0,0,16,0])
288 0461 THEN
289 0462 EXITLOOP; ! then stop looping
290 0463
291 0464 IF .exec_type NEQ adj$sc_pty_area ! If we are not an area router,
```



```
292 0465 3 THEN
293 0466 4 BEGIN
294 0467 4 BIND
295 0468 4 next_hop_addr = buffer [3*4,0,32,0];
296 0469 4
297 0470 4 node_name [0] = 32; ! Make descriptor of output buffer
298 0471 4 node_name [1] = node_name_buffer;
299 0472 4 node_name [0] = ! Get node name of next hop
300 0473 4 get_node_name(.next_hop_addr, node_name);
301 0474 4
302 0475 4 SELECTONEU .exec_type OF
303 0476 4 SET
304 0477 4 [adj$c_pty_ph4n,adj$c_pty_ph3n]:
305 0478 4 BEGIN
306 0479 4 write_line(%ASCII 'This is a nonrouting node, and does not have any network information. ');
307 0480 4 IF .next_hop_addr NEQ -1
308 0481 4 THEN
309 0482 4 write_line(%ASCII 'The designated router for !AS is node !AS !AS.',
310 0483 4 exec_name,
311 0484 4 format_nodeadr(.next_hop_addr),
312 0485 4 node_name);
313 0486 4 END;
314 0487 4 [OTHERWISE]:
315 0488 4 BEGIN
316 0489 4 IF .next_hop_addr NEQ -1
317 0490 4 THEN
318 0491 4 write_line(%ASCII 'The next hop to the nearest area router is node !AS !AS.',
319 0492 4 format_nodeadr(.next_hop_addr),
320 0493 4 node_name);
321 0494 4 END;
322 0495 4 TES;
323 0496 4 total_count = 1; ! Force some spacing afterwards
324 0497 4 EXITLOOP; ! Do not display area database
325 0498 4 END;
326 0499 4
327 0500 4 IF .total_count EQL 0 ! If first time through,
328 0501 4 THEN ! Print header line
329 0502 4 write_line(%ASCII '!/!13* Area Cost Hops Next Hop to Area!/');
330 0503 4
331 0504 4 buffer_ptr = buffer; ! Point to first node in buffer.
332 0505 4 buffer_count = .keys [0,0,32,0]; ! Get number of nodes returned in the
333 0506 4 ! buffer.
334 0507 4 WHILE .buffer_count GTR 0
335 0508 4 DO
336 0509 4 BEGIN
337 0510 4 buffer_ptr = format_area_info (.buffer_ptr);
338 0511 4 total_count = .total_count + 1; ! Increment # areas reachable
339 0512 4 buffer_count = .buffer_count - 1;
340 0513 4 END;
341 0514 4 END;
342 0515 4
343 0516 4 !
344 0517 4 ! As long as we aren't an endnode, display reachable nodes
345 0518 4 !
346 0519 4 !
347 0520 4 IF .exec_type NEQ adj$c_pty_ph4n ! If we aren't an endnode,
348 0521 4 AND .exec_type NEQ adj$c_pty_ph3n
```

```
0522 2 THEN
0523 BEGIN
0524 IF .total_count GTR 0
0525 THEN
0526 write_line(ASCII ' ');
0527
0528 display_nodes();
0529 END;
0530
0531
0532 Cleanup channel to ACP
0533
0534
0535 $DASSGN(CHAN = .channel);
0536
0537 RETURN;
0538
0539 1 END;
```

```
! If displayed at least 1 area,
! put 1 blank line here
! Display reachable nodes in our area

! Deassign the ACP channel
! Return to CLI dispatcher
```

```
.TITLE SHOW$NETWORK
.IDENT \V04-000\

.PSECT $SPLITS,NOWRT,NOEXE,2

00 00 00 3A 54 45 4E 5F 00000 P.AAB: .ASCII \ NET:\<0><0><0>
010E0005 00008 P.AAA: .LONG 17694725
00000000 0000C .ADDRESS P.AAB
01020041 0101001A 01010010 00010 P.AAC: .LONG 16842768, 16842778, 16908353
0001C P.AAE: .ASCII \VAX/VMS Network status for local node !A\
6B 72 6F 77 74 65 4E 20 53 4D 56 2F 58 41 56 0002B
63 6F 6C 20 72 6F 66 20 73 75 74 61 74 73 20 0003A
44 25 21 20 6E 6F 20 53 41 21 20 53 00044
010E0034 00050 P.AAD: .ASCII \S !AS on !XD\
00000000 00054 .LONG 17694772
00058 P.AAG: .ADDRESS P.AAE
010E0000 00058 P.AAF: .BLKB 0
00000000 0005C .LONG 17694720
14020041 14010013 14010012 14010011 14010010 0005C .ADDRESS P.AAG
00060 P.AAH: .LONG 335609872, 335609873, 335609874, -
335609875, 335675457
6F 72 6E 6F 6E 20 61 20 73 69 20 73 69 68 54 00074 P.AAJ: .ASCII \This is a nonrouting node, and does not \
64 6E 61 20 2C 65 64 6F 6E 20 67 6E 69 74 75 00083
20 74 6F 6E 20 73 65 6F 64 20 00092
72 6F 77 74 65 6E 20 79 6E 61 20 65 76 61 68 0009C
00 2E 6E 6F 69 74 61 6D 72 6F 66 6E 69 20 68 000AB
00 000BA
00 000BB
010E0045 000BC P.AAI: .ASCII <0>
00000000 000C0 .LONG 17694789
000C4 P.AAL: .ADDRESS P.AAJ
000D3 P.AAL: .ASCII \The designated router for !AS is node !A\
000E2
000EC
010E002E 000F4 P.AAK: .ASCII \S !AS.\<0><0>
00000000 000F8 .LONG 17694766
000FC P.AAN: .ADDRESS P.AAL
72 61 20 74 73 65 72 61 65 6E 20 65 68 74 20 0010B .ASCII \The next hop to the nearest area router \
```

CLRL - (SP)

0302



			60	AE	9F	00082	PUSHAB	KEY_DESC		
			98	AD	9F	00085	PUSHAB	NFB_DESC		
				7E	7C	00088	CLRQ	-(SP)		
			90	AD	9F	0008A	PUSHAB	IOSB		
				3B	DD	0008D	PUSHL	#56		
		7E	0000'	CF	3C	0008F	MOVZWL	CHANNEL, -(SP)		
				7E	D4	00094	CLRL	-(SP)		
		6B		0C	FB	00096	CALLS	#12, SYSSQIOW		
		56		50	DD	00099	MOVL	R0, STATUS		
		07		56	E9	0009C	BLBC	STATUS, 38		0384
		56	90	AD	3C	0009F	MOVZWL	IOSB, STATUS		0385
		13		56	E8	000A3	BLBS	STATUS, 68		
	0000007C	8F		56	D1	000A6	CMPL	STATUS, #124		0388
				85	11	000AD	BRB	18		
				56	DD	000AF	PUSHL	STATUS		0390
	00000000G	00		01	FB	000B1	CALLS	#1, LIBSSIGNAL		
					04	000B8	RET			0387
		57	00AC	CE	7D	000B9	MOVQ	BUFFER, EXEC_ADDR		0394
		6E	00B4	CE	3C	000BE	MOVZWL	BUFFER+8, EXEC_NAME		0396
		AE	08	AE	9E	000C3	MOVAB	EXEC_NAME+4, BUFFER, EXEC_NAME+4		0397
04	BE	00B6		6E	28	000C8	MOVCS	EXEC_NAME+4, BUFFER+10, #EXEC_NAME+4		0398
				7E	D4	000CF	CLRL	-(SP)		0404
			04	AE	9F	000D1	PUSHAB	EXEC_NAME		
				57	DD	000D4	PUSHL	EXEC_ADDR		0405
	0000V	CF		01	FB	000D6	CALLS	#1, FORMAT_NODEADR		
				50	DD	000DB	PUSHL	R0		
			48	A9	9F	000DD	PUSHAB	P.AAD		0404
		6A		04	FB	000E0	CALLS	#4, WRITE_LINE		
			50	A9	9F	000E3	PUSHAB	P.AAF		0408
		6A		01	FB	000E6	CALLS	#1, WRITE_LINE		
	00A4	CE	0200	8F	3C	000E9	MOVZWL	#512, BUFFER_DESC		0418
	00A8	CE	00AC	CE	9E	000F0	MOVAB	BUFFER, BUFFER_DESC+4		0419
	50	AE	48	8F	9A	000F7	MOVZBL	#72, KEY_DESC		0421
	54	AE	58	AE	9E	000FC	MOVAB	KEYS, KEY_DESC+4		0422
			58	AE	D4	00101	CLRL	KEYS		0424
	5C	AE		01	DD	00104	MOVL	#1, KEYS+4		0425
10			60	AE	B4	00108	CLRW	KEYS+8		0426
	00			00	2C	0010B	MOVCS	#0, (SP), #0, #16, NFB		0428
			AD	AD		00110				
	A2	AD		14	90	00112	MOVQ	#20, NFB+2		0431
	A0	AD	0222	8F	B0	00116	MOVW	#546, NFB		0430
	A4	AD	14000002	8F	DD	0011C	MOVL	#335544322, NFB+4		0433
			A3	AD	94	00124	CLRB	NFB+3		0434
	98	AD		24	DD	00127	MOVL	#36, NFB_DESC		0436
	9C	AD	AD	AD	9E	0012B	MOVAB	NFB, NFB_DESC+4		0437
B0	AD	58	A9	14	28	00130	MOVCS	#20, P.AXH, NFB+16		0445
				53	D4	00136	CLRL	TOTAL_COUNT		0447
				7E	7C	00138	CLRQ	-(SP)		0457
			00AC	CE	9F	0013A	PUSHAB	BUFFER_DESC		
				7E	D4	0013E	CLRL	-(SP)		
		60		AE	9F	00140	PUSHAB	KEY_DESC		
		98		AD	9F	00143	PUSHAB	NFB_DESC		
				7E	7C	00146	CLRQ	-(SP)		
		90		AD	9F	00148	PUSHAB	IOSB		
				3B	DD	0014B	PUSHL	#56		
		7E	0000'	CF	3C	0014D	MOVZWL	CHANNEL, -(SP)		
				7E	D4	00152	CLRL	-(SP)		

6B	0C	FB	00154	CALLS	#12, SYS\$QIOW	
56	50	DD	00157	MOVL	R0, STATUS	
79	56	E9	0015A	BLBC	STATUS, 11\$	0459
56	90	AD	3C 0015D	MOVZWL	IOSB, STATUS	0460
72	56	E9	00161	BLBC	STATUS, 11\$	
03	58	D1	00164	CMPL	EXEC_TYPE, #3	0464
	6F	13	00167	BEQL	12\$	
28	AE	20	DD 00169	MOVL	#32, NODE_NAME	0470
2C	AE	30	AE 9E 0016D	MOVAB	NODE_NAME_BUFFER, NODE_NAME+4	0471
		28	AE 9F 00172	PUSHAB	NODE_NAME	0473
52	00BC	CE	DD 00175	MOVL	NEXT_HOP_ADDR, R2	
		52	DD 0017A	PUSHL	R2	
0000V	CF	02	FB 0017C	CALLS	#2, GET_NODE_NAME	
28	AE	50	DD 00181	MOVL	R0, NODE_NAME	
01		58	D1 00185	CMPL	EXEC_TYPE, #1	0477
		05	13 00188	BEQL	8\$	
05		58	D1 0018A	CMPL	EXEC_TYPE, #5	
		28	12 0018D	BNEQ	9\$	
	00B4	C9	9F 0018F 8\$:	PUSHAB	P.AAI	0479
6A		01	FB 00193	CALLS	#1, WRITE_LINE	
FFFFFFF	8F	52	D1 00196	CMPL	R2, #-1	0480
		34	13 0019D	BEQL	10\$	
		28	AE 9F 0019F	PUSHAB	NODE_NAME	0482
		52	DD 001A2	PUSHL	R2	0484
0000V	CF	01	FB 001A4	CALLS	#1, FORMAT_NODEADR	
		50	DD 001A9	PUSHL	R0	
	08	AE	9F 001AB	PUSHAB	EXEC_NAME	0482
	00EC	C9	9F 001AE	PUSHAB	P.AAR	
6A		04	FB 001B2	CALLS	#4, WRITE_LINE	
		1C	11 001B5	BRB	10\$	0475
FFFFFFF	8F	52	D1 001B7 9\$:	CMPL	R2, #-1	0489
		13	13 001BE	BEQL	10\$	
		28	AE 9F 001C0	PUSHAB	NODE_NAME	0491
		52	DD 001C3	PUSHL	R2	0492
0000V	CF	01	FB 001C5	CALLS	#1, FORMAT_NODEADR	
		50	DD 001CA	PUSHL	R0	
	012C	C9	9F 001CC	PUSHAB	P.AAM	0491
6A		03	FB 001D0	CALLS	#3, WRITE_LINE	
53		01	DD 001D3 10\$:	MOVL	#1, TOTAL_COUNT	0496
		29	11 001D6 11\$:	BRB	16\$	0466
		53	D5 001D8 12\$:	TSTL	TOTAL_COUNT	0500
		07	12 001DA	BNEQ	13\$	
	0164	C9	9F 001DC	PUSHAB	P.AAO	0502
6A		01	FB 001E0	CALLS	#1, WRITE_LINE	
55	00AC	CE	9E 001E3 13\$:	MOVAB	BUFFER, BUFFER_PTR	0504
54	58	AE	DD 001E8	MOVL	KEYS, BUFFER_COUNT	0505
		03	14 001EC 14\$:	BGTR	15\$	0507
		FF47	31 001EE	BRW	7\$	
		55	DD 001F1 15\$:	PUSHL	BUFFER_PTR	0510
0000V	CF	01	FB 001F3	CALLS	#1, FORMAT_AREA_INFO	
55		50	DD 001F8	MOVL	R0, BUFFER_PTR	
		53	D6 001FB	INCL	TOTAL_COUNT	0511
		54	D7 001FD	DECL	BUFFER_COUNT	0512
		EB	11 001FF	BRB	14\$	0507
05		58	D1 00201 16\$:	CMPL	EXEC_TYPE, #5	0520
		15	13 00204	BEQL	18\$	
01		58	D1 00206	CMPL	EXEC_TYPE, #1	0521

SHOWNETWORK  
V04-000

B 9  
16-Sep-1984 00:39:09  
14-Sep-1984 12:09:32

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SHONET.B32;1

Page 12  
(3)

			10	13	00209	BEQ	188		
			53	D5	00208	TSTL	TOTAL_COUNT		0524
			07	15	0020D	BLEQ	178		
		016C	C9	9F	0020F	PUSHAB	P.AAQ		0526
			01	FB	00213	CALLS	#1, WRITE_LINE		
0000V	6A		00	FB	00216	CALLS	#0, DISPLAY_NODES		0528
	CF		CF	3C	00218	MOVZWL	CHANNEL, -(SP)		0535
00000000G	7E	0000'	01	FB	00220	CALLS	#1, SYSDASSGN		
	00		04		00227	RET			0539

; Routine Size: 552 bytes.      Routine Base: \$CODE\$ + 0000



```
368 0540 1 ROUTINE display_nodes: NOVALUE =
369 0541
370 0542
371 0543
372 0544 This routine displays all reachable nodes in our area.
373 0545
374 0546 Inputs:
375 0547
376 0548 None
377 0549
378 0550 Outputs:
379 0551
380 0552 None
381 0553
382 0554
383 0555
384 0556 BEGIN
385 0557
386 0558 LITERAL
387 0559 buffer_size = 512; ! Size of return buffer.
388 0560
389 0561 LOCAL
390 0562 nfb: BBLOCK [nfb$length+20*4], ! Network function block
391 0563 nfb_desc: VECTOR [2], ! Descriptor of NFB
392 0564 iosb: BBLOCK [8], ! I/O status block
393 0565 total_node_count, ! Number of nodes displayed
394 0566 buffer_node_count, ! Number of nodes returned in buffer
395 0567 buffer: BBLOCK [buffer_size], ! Return buffer
396 0568 buffer_desc: VECTOR [2], ! Descriptor of above buffer
397 0569 buffer_ptr, ! Pointer to return buffer
398 0570 keys: BBLOCK [4+8+nfb$ctx_size], ! Buffer for search keys & context
399 0571 key_desc: VECTOR [2], ! Descriptor of above buffer
400 0572 status;
401 0573
402 0574
403 0575 ! Display the cost/hops information for all nodes in this area
404 0576
405 0577
406 0578 buffer_desc [0] = buffer_size; ! Construct descriptor of return buffer
407 0579 buffer_desc [1] = buffer;
408 0580
409 0581 key_desc [0] = 4 + 8 + nfb$ctx_size; ! Longword overhead, TWO search values
410 0582 key_desc [1] = keys; ! and fixed context area
411 0583
412 0584 keys [0,0,32,0] = 0; ! Zero count of fields in P4 (unused)
413 0585 keys [4,0,32,0] = true; ! REA search value EQL TRUE
414 0586 keys [8,0,32,0] = true; ! LOO search value NEQ true
415 0587 keys [12,0,16,0] = 0; ! Start key = at beginning
416 0588
417 0589 CH$FILL(0,nfb$length,nfb); ! Pre-zero NFB fields
418 0590
419 0591 nfb [nfb$b_fct] = nfb$fc_show; ! Request "show" function
420 0592 nfb [nfb$b_database] = nfb$db_ndi; ! of node database
421 0593 nfb [nfb$b_flags] = nfb$mu[t]; ! Request multiple entries per Q10
422 0594 nfb [nfb$l_srch_key] = nfb$ndi_rea; ! Only return reachable nodes
423 0595 nfb [nfb$b_oper] = nfb$op_eq; ! by checking if field EQL P2 value
424 0596 nfb [nfb$l_srch2_key] = nfb$ndi_loo; ! Do not return "loop nodes"
```

```
425 0597 nfb [nfb$b_oper2] = nfb$c_op_neq; ! by checking if field NEQ P2 value
426 0598
427 0599 nfb_desc [0] = $BYTEOFFSET(nfb$l_fldid) + 8*4; ! Construct descriptor of NFB
428 0600 nfb_desc [1] = nfb;
429 0601
430 0602 CHSMOVE(8*4, UPLIT LONG( ! Request the following fields:
431 0603 nfb$c_ndi_tad, ! Translated node address
432 0604 nfb$c_ndi_acl, ! Active links
433 0605 nfb$c_ndi_dco, ! Destination cost
434 0606 nfb$c_ndi_dho, ! Destination hops
435 0607 nfb$c_ndi_nnd, ! Next hop node address
436 0608 nfb$c_ndi_nna, ! Node name
437 0609 nfb$c_ndi_nnn, ! Next hop node name
438 0610 nfb$c_ndi_dli, ! Destination circuit name
439 0611 nfb [nfb$l_fldid]);
440 0612
441 0613 total_node_count = 0; ! Initialize node count
442 0614
443 0615 WHILE true
444 0616 DO
445 0617 BEGIN
446 0618 status = $QIOW(FUNC = IOS$ACPCONTROL, ! Issue control function
447 0619 CHAN = .channel,
448 0620 IOSB = iosb,
449 0621 P1 = nfb_desc, ! Address of NDB descriptor
450 0622 P2 = key_desc, ! Address of key buffer descriptor
451 0623 P4 = buffer_desc); ! Address of return buffer descriptor
452 0624
453 0625 IF NOT .status ! If error detected,
454 0626 OR NOT (status = .iosb [0,0,16,0])
455 0627 THEN
456 0628 EXITLOOP; ! then stop looping
457 0629
458 0630 IF .total_node_count EQL 0 ! If first time through,
459 0631 THEN ! Print header line
460 0632 write_line(%ASCII '!!/!8* Node!9* Links Cost Hops Next Hop to Node!/'');
461 0633
462 0634 buffer_ptr = buffer; ! Point to first node in buffer.
463 0635 buffer_node_count = .keys [0,0,32,0]; ! Get number of nodes returned in the
464 0636 ! buffer.
465 0637 WHILE .buffer_node_count GTR 0
466 0638 DO
467 0639 BEGIN
468 0640 buffer_ptr = format_node_info (.buffer_ptr);
469 0641 total_node_count = ! Increment # nodes reachable
470 0642 .total_node_count + 1;
471 0643 buffer_node_count = .buffer_node_count - 1;
472 0644 END;
473 0645
474 0646 END;
475 0647
476 0648 IF .status EQL ss$endoffile ! If normal termination,
477 0649 THEN
478 0650 BEGIN
479 0651 IF .total_node_count GTR 1 ! If more than local node found,
480 0652 THEN ! Write the total
481 0653 write_line(%ASCII '!!/!16* Total of !UL node!%S.',
```

```
0654      .total_node_count);
0655      END
0656      ELSE
0657      BEGIN
0658      IF .status EQL ss$ devnotmount
0659      THEN SIGNAL(show$ nonet)
0660      ELSE SIGNAL(.status);
0661      END;
0662
0663      END;
```

```
! If ACP not yet started,
! then indicate network not up
! Else, report the status
```

```
02020043 02010022 02010018 02010017 02010014 02010010 0017C P.AAS: .LONG 33619984, 33619988, 33619991, 33619992, -
0202004D 02020059 00194 33620002, 33685571, 33685593, 33685581
20 20 2A 39 21 65 64 6F 4E 20 2A 38 21 2F 21 0019C P.AAU: .ASCII \!/:8* Node!9* Links Cost Hops Next \
6F 48 20 20 74 73 6F 43 20 20 73 68 6E 69 4C 001AB
00 00 2F 21 65 64 6F 4E 20 6F 74 20 70 6F 48 001BA
001C4
001D3
010E0035 001D4 P.AAT: .LONG 17694773
00000000 001D8 .ADDRESS P.AAU
66 6F 20 6C 61 74 6F 54 20 2A 36 31 21 2F 21 001DC P.AAW: .ASCII \!/:16* Total of !UL node!%S.\
2E 53 25 21 65 64 6F 6E 20 4C 55 21 20 001EB
010E001C 001F8 P.AAV: .LONG 17694748
00000000 001FC .ADDRESS P.AAW
```

```
.PSECT $CODE$,NOWRT,2
```

```
003C 00000 DISPLAY_NODES:
```

```
Save R2,R3,R4,R5
MOVAB -712(SP), SP
MOVZWL #512, BUFFER_DESC
MOVAB BUFFER, BUFFER_DESC+4
MOVZBL #76, KEY_DESC
MOVAB KEYS, KEY_DESC+4
CLRL KEYS
MOVL #1, KEYS+4
MOVL #1, KEYS+8
CLRW KEYS+12
MOVCS #0, (SP), #0, #16, NFB
MOVAB #2, NFB+2
MOVW #546, NFB
MOVL #33554435, NFB+4
CLRB NFB+3
MOVL #33554434, NFB+8
MOVAB #3, NFB+12
MOVL #48, NFB_DESC
MOVAB NFB, NFB_DESC+4
MOVCS #32, P.AXS, NFB+16
CLRL TOTAL_NODE_COUNT
CLRQ -(SP)
```

```
0540
0578
0579
0581
0582
0584
0585
0586
0587
0589
0592
0591
0594
0595
0596
0597
0599
0600
0611
0613
0623
```



		5C	AE	9F	00065	PUSHAB	BUFFER_DESC	
		7E	D4	00068	CLRL	-(SP)		
		10	AE	9F	0006A	PUSHAB	KEY_DESC	
		98	AD	9F	0006D	PUSHAB	NFB_DESC	
		7E	7C	00070	CLRL	-(SP)		
		90	AD	9F	00072	PUSHAB	IOSB	
		38	DD	00075	PUSHL	#56		
	7E	0000'	CF	3C	00077	MOVZWL	CHANNEL, -(SP)	
			7E	D4	0007C	CLRL	-(SP)	
00000000G	00		OC	FB	0007E	CALLS	#12, SYSSQIOW	
	52		50	D0	00085	MOVL	RO, STATUS	
	2E		52	E9	00088	BLBC	STATUS, 4\$	0625
	52	90	AD	3C	0008B	MOVZWL	IOSB, STATUS	0626
	27		52	E9	0008F	BLBC	STATUS, 4\$	
			53	D5	00092	TSTL	TOTAL_NODE_COUNT	0630
		0000'	09	12	00094	BNEQ	2\$	
			CF	9F	00096	PUSHAB	P.AAT	0632
0000V	CF		01	FB	0009A	CALLS	#1, WRITE_LINE	
	55	5C	AE	9E	0009F	MOVAB	BUFFER, BOFFER_PTR	0634
	54	08	AE	D0	000A3	MOVL	KEYS, BUFFER_NODE_COUNT	0635
			BA	15	000A7	BLEQ	1\$	0637
			55	DD	000A9	PUSHL	BUFFER_PTR	0640
0000V	CF		01	FB	000AB	CALLS	#1, FORMAT_NODE_INFO	
	55		50	D0	000B0	MOVL	RO, BUFFER_PTR	
			53	D6	000B3	INCL	TOTAL_NODE_COUNT	0642
			54	D7	000B5	DECL	BUFFER_NODE_COUNT	0643
			EE	11	000B7	BRB	3\$	0637
00000870	8F		52	D1	000B9	CMPL	STATUS, #2160	0648
			11	12	000C0	BNEQ	5\$	
	01		53	D1	000C2	CMPL	TOTAL_NODE_COUNT, #1	0651
			26	15	000C5	BLEQ	8\$	
		0000'	53	DD	000C7	PUSHL	TOTAL_NODE_COUNT	0654
			CF	9F	000C9	PUSHAB	P.AAV	0653
0000V	CF		02	FB	000CD	CALLS	#2, WRITE_LINE	
				04	000D2	RET		0648
0000007C	8F		52	D1	000D3	CMPL	STATUS, #124	0658
			08	12	000DA	BNEQ	6\$	
		00000000G	8F	DD	000DC	PUSHL	#SHOW\$_NONET	0659
			02	11	000E2	BRB	7\$	
			52	DD	000E4	PUSHL	STATUS	0660
00000000G	00		01	FB	000E6	CALLS	#1, LIB\$SIGNAL	
				04	000ED	RET		0663

; Routine Size: 238 bytes, Routine Base: \$CODE\$ + 0228

```
493 0664 ROUTINE format_area_info (info_ptr: REF VECTOR) =
494 0665
495 0666
496 0667
497 0668 This routine accepts a pointer to one area's information in the buffer
498 0669 returned by NETACP. It formats this information and writes it to the
499 0670 output stream.
500 0671
501 0672 Inputs:
502 0673
503 0674 info_ptr = Address of the beginning of the area's information in
504 0675 the buffer returned by NETACP.
505 0676
506 0677 Outputs:
507 0678
508 0679 Routine value = Address of next byte beyond area's information.
509 0680
510 0681
511 0682 BEGIN
512 0683
513 0684 LOCAL
514 0685 ptr: REF BBLOCK, ! Pointer into area information.
515 0686 circ_name: VECTOR [2], ! Descriptor of circuit name
516 0687 next_hop_name_buffer: VECTOR [32,BYTE], ! Buffer to hold next hop name
517 0688 next_hop_name: VECTOR [2]; ! Descriptor of next hop node name
518 0689
519 0690 next_hop_name [0] = 32; ! Make descriptor of output buffer
520 0691 next_hop_name [1] = next_hop_name_buffer;
521 0692 next_hop_name [0] =
522 0693 get_node_name(.info_ptr [3], next_hop_name);
523 0694
524 0695 ptr = info_ptr [4]; ! Point to word-counted circuit name
525 0696
526 0697 circ_name [0] = .ptr [0,0,16,0]; ! Construct descriptor of circuit name
527 0698 circ_name [1] = .ptr + 2;
528 0699 ptr = .ptr + 2 + .ptr [0,0,16,0]; ! Skip by string in buffer
529 0700
530 0701
531 0702 Output the line
532 0703
533 0704
534 0705 write_line(%ASCII '!13* !3UL !4UL !4UL !10AS-> !6AS !AS',
535 0706 .info_ptr [0], ! Area number
536 0707 .info_ptr [1], ! Least cost to area
537 0708 .info_ptr [2], ! Actual hops to area
538 0709 (IF .circ_name [0] EQL 0 then %ASCII '(Local)' ELSE circ_name), ! Circuit name
539 0710 format_nodeadr(.info_ptr [3]), ! Next hop node address
540 0711 next_hop_name); ! Next hop node name
541 0712
542 0713 RETURN .ptr; ! Return updated pointer
543 0714
544 0715 END;
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

```
55 34 21 20 20 20 4C 55 33 21 20 2A 33 31 21 00200 P.AAY: .ASCII \!13+ !3UL !4UL !4UL !10AS-> !6AS\
31 21 20 20 20 20 53 41 36 21 20 3E 2D 53 41 30 0020F
                                0021E
                                0022B
                                0022C P.AAX: .ASCII \!AS\
                                00230 .LONG 17694764
                                00234 .ADDRESS P.AAY
00 29 6C 61 63 6F 4C 28 00234 P.ABA: .ASCII \!(Local)\<0>
                                0023C P.AAZ: .LONG 17694727
                                00240 .ADDRESS P.ABA
```

.PSECT \$CODE\$,NOWRT,2

000C 00000 FORMAT\_AREA INFO:

SE	2C	C2	00002	.WORD	Save R2,R3	0664	
	20	DD	00005	SUBL2	#44, SP		
04 AE	08	AE	9E	00007	PUSHL	#32	0690
		5E	DD	0000C	MOVAB	NEXT_HOP_NAME_BUFFER, NEXT_HOP_NAME+4	0691
53	04	AC	DD	0000E	PUSHL	SP	0693
	0C	A3	DD	00012	MOVL	INFO_PTR, R3	
0000V CF		02	FB	00015	PUSHL	12(R3)	
6E		50	DD	0001A	CALLS	#2, GET_NODE_NAME	
52	10	A3	9E	0001D	MOVL	R0, NEXT_HOP_NAME	
50		62	3C	00021	MOVAB	16(R3), PTR	0695
28 AE		50	DD	00024	MOVZWL	(PTR), R0	0697
2C AE	02	A2	9E	00028	MOVL	R0, CIRC_NAME	
52	02	A042	9E	0002D	MOVAB	2(R2), CIRC_NAME+4	0698
		5E	DD	00032	MOVAB	2(R0)[PTR], -PTR	0699
	0C	A3	DD	00034	PUSHL	SP	0705
0000V CF		01	FB	00037	PUSHL	12(R3)	0710
		50	DD	0003C	CALLS	#1, FORMAT_NODEADR	
	30	AE	D5	0003E	PUSHL	R0	
		07	12	00041	TSTL	CIRC_NAME	0709
50	0000'	CF	9E	00043	BNEQ	1\$	
		04	11	00048	MOVAB	P.AAZ, R0	
50	30	AE	9E	0004A	BRB	2\$	
		50	DD	0004E	MOVAB	CIRC_NAME, R0	
7E	04	A3	7D	00050	PUSHL	R0	
		63	DD	00054	MOVQ	4(R3), -(SP)	0707
	0000'	CF	9F	00056	PUSHL	(R3)	0706
0000V CF		07	FB	0005A	PUSHAB	P.AAX	0705
50		52	DD	0005F	CALLS	#7, WRITE_LINE	
		04	00062	RETL	MOVL	PTR, R0	0713
					RET		0715

; Routine Size: 99 bytes. Routine Base: \$CODE\$ + 0316



```
546 0716 1 ROUTINE format_node_info (info_ptr: REF VECTOR) =
547 0717
548 0718 --
549 0719
550 0720 This routine accepts a pointer to one node's information in the buffer
551 0721 returned by NETACP. It formats this information and writes it to the
552 0722 output stream.
553 0723
554 0724 Inputs:
555 0725
556 0726 info_ptr = Address of the beginning of the node's information in
557 0727 the buffer returned by NETACP.
558 0728
559 0729 Outputs:
560 0730
561 0731 Routine value = Address of next byte beyond node's information.
562 0732 --
563 0733
564 0734 BEGIN
565 0735
566 0736 LOCAL
567 0737 ptr: REF BBLOCK, ! Pointer into node information.
568 0738 node_name: VECTOR [2], ! Descriptor of node name
569 0739 circ_name: VECTOR [2], ! Descriptor of circuit name
570 0740 next_hop_name: VECTOR [2], ! Descriptor of next hop node name
571 0741 next_hop_ptr: REF VECTOR [2], ! Ptr to formatted next hop descriptor
572 0742 next_hop_addr_buffer: VECTOR [32,BYTE], ! Buffer to hold next hop address
573 0743 next_hop_addr: VECTOR [2]; ! Descriptor of next hop node address
574 0744
575 0745 ptr = info_ptr [5]; ! Point to word-counted node name
576 0746
577 0747 node_name [0] = .ptr [0,0,16,0]; ! Construct descriptor of node name
578 0748 node_name [1] = .ptr + 2;
579 0749 ptr = .ptr + 2 + .ptr [0,0,16,0]; ! Skip by string in buffer
580 0750
581 0751 next_hop_name [0] = .ptr [0,0,16,0]; ! Construct descriptor of next hop
582 0752 next_hop_name [1] = .ptr + 2;
583 0753 ptr = .ptr + 2 + .ptr [0,0,16,0]; ! Skip by string in buffer
584 0754
585 0755 circ_name [0] = .ptr [0,0,16,0]; ! Construct descriptor of circuit name
586 0756 circ_name [1] = .ptr + 2;
587 0757 ptr = .ptr + 2 + .ptr [0,0,16,0]; ! Skip by string in buffer
588 0758
589 0759 next_hop_ptr = format_nodeadr(.info_ptr [4]); ! Format next hop address
590 0760 next_hop_addr [0] = .next_hop_ptr [0]; ! Save descriptor of formatted string
591 0761 next_hop_addr [1] = next_hop_addr_buffer;
592 0762 CHSMOVE(.next_hop_ptr [0], .next_hop_ptr [1], .next_hop_addr [1]);
593 0763
594 0764
595 0765 ! Output the line
596 0766
597 0767
598 0768 write_line(%ASCII '!'4* !15<!6AS !AS!> !6UL !4UL !4UL !10AS-> !6AS !AS',
599 0769 format_nodeadr(.info_ptr [0]), ! Node address
600 0770 node_name, ! Node name
601 0771 (IF .info_ptr [1] GEQ 0 THEN .info_ptr [1] ELSE 0), ! Active Links
602 0772 .info_ptr [2], ! Destination cost
```

```

: 603      0773 2      .info_ptr [3],      ! Destination hops
: 604      0774 2      (IF .circ_name [0] EQL 0 then %ASCII (Local) ELSE circ_name), ! Circuit name
: 605      0775 2      next_hop_addr,      ! Next hop node address
: 606      0776 2      next_hop_name);      ! Next hop node name
: 607      0777 2
: 608      0778 2      RETURN .ptr;      ! Return updated pointer
: 609      0779 2
: 610      0780 1      END;
```

```

                                .PSECT $SPLITS,NOWRT,NOEXE,2
41 21 20 53 41 36 21 3C 35 31 21 20 2A 34 21 00244 P.ABC: .ASCII \!4* !15<!6AS !AS!> !6UL !4UL !4UL \
20 4C 55 34 21 20 20 4C 55 36 21 20 3E 21 53 00253
41 21 20 53 41 36 21 20 20 20 20 4C 55 34 21 20 00262
                                .ASCII \!10AS-> !6AS !AS\
                                010E0038 0027C P.ABB: .LONG 17694776
                                00000000 00280 .ADDRESS P.ABC
00 29 6C 61 63 6F 4C 28 00284 P.ABE: .ASCII \((Local)\<0>
                                010E0007 0028C P.ABD: .LONG 17694727
                                00000000 00290 .ADDRESS P.ABE
```

.PSECT \$CODE\$,NOWRT,2

00FC 00000 FORMAT\_NODE INFO:

```

                                .WORD Save R2,R3,R4,R5,R6,R7
                                MOVAB -64(SP), SP
                                MOVL INFO_PTR, R7
                                MOVAB 20(R7), PTR
                                MOVZWL (PTR), R0
                                MOVL R0, NODE_NAME
                                MOVAB 2(R6), NODE_NAME+4
                                MOVAB 2(R0)(PTR), -PTR
                                MOVZWL (PTR), R0
                                MOVL R0, NEXT_HOP_NAME
                                MOVAB 2(R6), NEXT_HOP_NAME+4
                                MOVAB 2(R0)(PTR), -PTR
                                MOVZWL (PTR), R0
                                MOVL R0, CIRC_NAME
                                MOVAB 2(R6), CIRC_NAME+4
                                MOVAB 2(R0)(PTR), -PTR
                                PUSHL 16(R7)
                                CALLS #1, FORMAT_NODEADR
                                MOVL (NEXT_HOP_PTR), NEXT_HOP_ADDR
                                MOVAB NEXT_HOP_ADDR_BUFFER, NEXT_HOP_ADDR+4
                                MOVC3 (NEXT_HOP_PTR), 24(NEXT_HOP_PTR), -
                                @NEXT_HOP_ADDR+4
                                PUSHAB NEXT_HOP_NAME
                                PUSHAB NEXT_HOP_ADDR
                                TSTL CIRC_NAME
                                BNEQ 1$
                                MOVAB P.ABD, R0
                                BRB 2$
                                28 AE 9F 00057 PUSHAB NEXT_HOP_NAME
                                04 AE 9F 0005A PUSHAB NEXT_HOP_ADDR
                                38 AE D5 0005D TSTL CIRC_NAME
                                07 12 00060 BNEQ 1$
                                50 0000 CF 9E 00062 MOVAB P.ABD, R0
                                04 11 00067 BRB 2$
```

0716  
0745  
0747  
0748  
0749  
0751  
0752  
0753  
0755  
0756  
0757  
0759  
0760  
0761  
0762  
0768  
0774

SHOW\$NETWORK  
V04-000

K 9  
16-Sep-1984 00:39:09 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:09:32 [CLIUTL.SRC]SHONET.B32;1

Page 21  
(6)

50	38	AE	9E	00069	1\$:	MOVAB	CIRC_NAME, R0	
		50	DD	0006D	2\$:	PUSHL	R0	
7E	08	A7	7D	0006F		MOVQ	8(R7), -(SP)	0772
	04	A7	D5	00073		TSTL	4(R7)	0771
		05	19	00076		BLSS	3\$	
	04	A7	DD	00078		PUSHL	4(R7)	
		02	11	0007B		BRB	4\$	
		7E	D4	0007D	3\$:	CLRL	-(SP)	
	50	AE	9F	0007F	4\$:	PUSHAB	NODE_NAME	0768
		67	DD	00082		PUSHL	(R7)	0769
0000V	CF	01	FB	00084		CALLS	#1, FORMAT_NODEADR	
		50	DD	00089		PUSHL	R0	
		CF	9F	0008B		PUSHAB	P, ABB	0768
0000V	CF	09	FB	0008F		CALLS	#9, WRITE_LINE	
	50	56	D0	00094		MOVL	PTR, R0	0778
		04	00	00097		RET		0780

; Routine Size: 152 bytes. Routine Base: \$CODE\$ + 0379



```
612 0781 1 ROUTINE get_node_name (addr, buffer_desc: REF VECTOR) =
613 0782 1
614 0783 1 ---
615 0784 1
616 0785 1 This routine returns the node name associated with a given node
617 0786 1 address.
618 0787 1
619 0788 1 Inputs:
620 0789 1
621 0790 1 addr = Node address
622 0791 1 buffer_desc = Address of descriptor of output buffer
623 0792 1
624 0793 1 Outputs:
625 0794 1
626 0795 1 Routine Value = Length of returned string
627 0796 1
628 0797 1
629 0798 2 BEGIN
630 0799 2
631 0800 2 LOCAL
632 0801 2 nfb: BBLOCK [nfb$length+1*4], ! Network function block
633 0802 2 nfb_desc: VECTOR [2], ! Descriptor of NFB
634 0803 2 iosb: BBLOCK [8], ! I/O status block
635 0804 2 keys: BBLOCK [4+4+nfb$ctx_size], ! Buffer for search keys & context
636 0805 2 key_desc: VECTOR [2], ! Descriptor of above buffer
637 0806 2 buffer: BBLOCK [16], ! P4 buffer (for node name)
638 0807 2 p4_desc: VECTOR [2], ! Descriptor of above buffer
639 0808 2 status;
640 0809 2
641 0810 2 key_desc [0] = 4 + 4 + nfb$ctx_size; ! Longword overhead, ONE search value
642 0811 2 key_desc [1] = keys; ! and fixed context area
643 0812 2
644 0813 2 keys [0,0,32,0] = 0; ! Zero count of fields in P4 (unused)
645 0814 2 keys [4,0,32,0] = addr; ! Insert desired node address
646 0815 2 keys [8,0,16,0] = 0; ! Start key = at beginning
647 0816 2
648 0817 2 p4_desc [0] = 16; ! Setup descriptor of P4 buffer
649 0818 2 p4_desc [1] = buffer;
650 0819 2
651 0820 2 CH$FILL(0,nfb$length,nfb); ! Pre-zero NFB fields
652 0821 2
653 0822 2 nfb [nfb$b_fct] = nfb$fc_show; ! Request 'show' function
654 0823 2 nfb [nfb$b_database] = nfb$db_ndi; ! of node database
655 0824 2 nfb [nfb$sl_srch_key] = nfb$ndi_tad; ! Search for matching address
656 0825 2 nfb [nfb$b_oper] = nfb$op_eq; ! using 'EQL' comparison
657 0826 2
658 0827 2 nfb_desc [0] = $BYTEOFFSET(nfb$sl_fldid) + 1*4; ! Construct descriptor of NFB
659 0828 2 nfb_desc [1] = nfb;
660 0829 2
661 0830 2 CH$MOVE(1*4, UPLIT LONG( ! Request the following fields:
662 0831 2 nfb$ndi_nna), ! Node name
663 0832 2 nfb [nfb$sl_fldid]);
664 0833 2
665 P 0834 2 status = $QIOW(FUNC = IOS$ACPCONTROL, ! Issue control function
666 P 0835 2 CHAN = channel,
667 P 0836 2 IOSB = iosb,
668 P 0837 2 P1 = nfb_desc, ! Address of NDB descriptor
```

```

669      P 0838      2      P2 = key_desc,
670      0839      2      P4 = p4_desc);
671      0840      2
672      0841      2      IF NOT .status
673      0842      2      OR NOT (status = .iosb [0,0,16,0])
674      0843      2      THEN
675      0844      2      RETURN 0
676      0845      2      ELSE
677      0846      2      BEGIN
678      0847      2      CH$MOVE(.buffer [0,0,16,0], buffer [2,0,0,0], .buffer_desc [1]);
679      0848      2      RETURN .buffer [0,0,16,0];
680      0849      2      END;
681      0850      2
682      0851      2      END;

```

```

! Address of key buffer descriptor
! Address of return buffer descriptor
! If error detected,
! Return null string
! Return length of string

```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

02020043 00294 P.ABF: .LONG 33685571

.PSECT \$CODE\$,NOWRT,2

003C 00000 GET\_NODE\_NAME:

			5E	FF78	CE	9E	00002	WORD	Save R2,R3,R4,R5	
			14	AE	48	8F	9A	00007	MOVAB	-136(SP), SP
			18	AE	1C	AE	9E	0000C	MOVZBL	#72, KEY_DESC
					1C	AE	D4	00011	MOVAB	KEYS, KEY_DESC+4
			20	AE	04	AC	D0	00014	CLRL	KEYS
					24	AE	B4	00019	MOVL	ADDR, KEYS+4
						10	DD	0001C	CLRW	KEYS+8
			04	AE	08	AE	9E	0001E	PUSHL	#16
10	00		6E		00	2C	00023	MOVAB	BUFFER, P4_DESC+4	
				78	AE		00028	MOVCS	#0, (SP), #0, #16, NFB	
			78	AE	22	90	0002A	MOVAB	#34, NFB	
			7A	AE	02	90	0002E	MOVAB	#2, NFB+2	
			7C	AE	02010010	8F	D0	00032	MOVL	#33619984, NFB+4
					7B	AE	94	0003A	CLRB	NFB+3
			70	AE		14	D0	0003D	MOVL	#20, NFB_DESC
			74	AE	78	AE	9E	00041	MOVAB	NFB, NFB_DESC+4
			FC	AD	0000	CF	D0	00046	MOVL	P.ABF, NFB+16
						7E	7C	0004C	CLRW	-(SP)
				08	AE	9F	0004E	PUSHAB	P4_DESC	
					7E	D4	00051	CLRL	-(SP)	
				28	AE	9F	00053	PUSHAB	KEY_DESC	
				E4	AD	9F	00056	PUSHAB	NFB_DESC	
					7E	7C	00059	CLRW	-(SP)	
				DC	AD	9F	0005B	PUSHAB	IOSB	
					38	DD	0005E	PUSHL	#56	
			7E	0000	CF	3C	00060	MOVZWL	CHANNEL, -(SP)	
					7E	D4	00065	CLRL	-(SP)	
		00000000G	00		0C	FB	00067	CALLS	#12, SYS\$QIOW	
			07		50	E9	0006E	BLBC	STATUS, 18	
			50	68	AE	3C	00071	MOVZWL	IOSB, STATUS	

07B1  
0810  
0811  
0813  
0814  
0815  
0817  
0818  
0820  
0822  
0823  
0824  
0825  
0827  
0828  
0830  
0839  
0841  
0842

SHOWSNETWORK  
V04-000

N 9  
16-Sep-1984 00:39:09 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:09:32 [CLIUTL.SRC]SHONET.B32;1

Page 24  
(7)

		03		50	E8	00075		BLBS	STATUS, 2\$	
				50	D4	00078	1\$:	CLRL	RO	
					04	0007A		RET		
		50	08	AC	D0	0007B	2\$:	MOVL	BUFFER_DESC, RO	
04	B0	0A	08	AE	28	0007F		MOVC3	BUFFER, BUFFER+2, 24(RO)	
		50	08	AE	3C	00086		MOVZWL	BUFFER, RO	
					04	0008A		RET		

: 0846  
: 0847  
: 0848  
: 0851

; Routine Size: 139 bytes, Routine Base: \$CODE\$ + 0411



```

: 684 0852 1 ROUTINE write_line (message, args): NOVALUE =
: 685 0853 1
: 686 0854 1 ---
: 687 0855 1
: 688 0856 1 This routine accepts a control string and a series of FA0
: 689 0857 1 arguments, and writes the resulting line to the output stream.
: 690 0858 1
: 691 0859 1 Inputs:
: 692 0860 1
: 693 0861 1 message = Message control string
: 694 0862 1 args = First FA0 argument (any number of arguments may follow)
: 695 0863 1
: 696 0864 1 Outputs:
: 697 0865 1
: 698 0866 1 None
: 699 0867 1 ---
: 700 0868 1
: 701 0869 2 BEGIN
: 702 0870 2
: 703 0871 2 show$write_line(.message, args); ! Use standard SHOW output routine
: 704 0872 2
: 705 0873 1 END;

```

```

0000 00000 WRITE_LINE:
08 AC 9F 00002 .WORD Save nothing
04 AC DD 00005 PUSHAB ARGS
02 FB 00008 PUSHL MESSAGE
04 0000D CALLS #2, SHOW$WRITE_LINE
RET
0000G CF

```

```

: 0852
: 0871
:
: 0873

```

; Routine Size: 14 bytes, Routine Base: \$CODE\$ + 049C

```
0874 1 ROUTINE format_nodeadr(address) =
0875 1
0876 1 ---
0877 1
0878 1 This routine formats a 16-bit node address into an
0879 1 formatted ASCII string of the form <area>.<node>.
0880 1 If the area number is zero, then the area portion
0881 1 is omitted.
0882 1
0883 1 Inputs:
0884 1
0885 1 address = 16-bit node address
0886 1
0887 1 Outputs:
0888 1
0889 1 Routine = Address of descriptor of string describing address
0890 1
0891 1 Since the string & descriptor is stored in OWN storage, it must
0892 1 be copied immediately after returning (with a standard routine
0893 1 such as "append").
0894 1 ---
0895 1
0896 2 BEGIN
0897 2
0898 2 OWN
0899 2 string: VECTOR [40,BYTE], ! Formatted node address string
0900 2 desc: VECTOR [2]; ! FAO result string descriptor
0901 2
0902 2 desc [0] = 40; ! Setup descriptor for FAO
0903 2 desc [1] = string;
0904 2
0905 2 IF .address <10,6,0> EQL 0 ! If area = 0,
0906 2 THEN ! Format only node
0907 2 $FAO(XASCID ' !UL',
0908 2 desc, desc,
0909 2 .address)
0910 2 ELSE ! Format area and node
0911 2 $FAO(XASCID '!2UL.!UL',
0912 2 desc, desc
0913 2 .address <10,6,0>,
0914 2 .address <0,10,0>);
0915 2
0916 2 RETURN desc;
0917 2
0918 1 END;
```

```
00 00 4C 55 21 20 20 20 00298 P.ABH: .ASCII \ !UL\<0><0>
010E0006 002A0 P.ABG: .LONG 17694726
00000000 002A4 .ADDRESS P.ABH
4C 55 21 2E 4C 55 32 21 002A8 P.ABJ: .ASCII \!2UL.!UL\
010E0008 002B0 P.ABI: .LONG 17694728
00000000 002B4 .ADDRESS P.ABJ
```

.PSECT \$PLITS,NOWRT,NOEXE,2

```
.PSECT $OWN$,NOEXE,2
00002 .BLKB 2
00004 STRING: .BLKB 40
0002C DESC: .BLKB 8
.EXTRN SYS$FAO
.PSECT $CODE$,NOWRT,2
```

				000C	00000	FORMAT	NODEADR:	
	53	00000000G	00	9E	00002	.WORD	Save R2,R3	0874
	52	0000'	CF	9E	00009	MOVAB	SYS\$FAO, R3	
	62		28	DD	0000E	MOVAB	DESC, R2	0902
04	A2	D8	A2	9E	00011	MOVL	#40, DESC	0903
FC	8F	05	AC	93	00016	BITB	ADDRESS+1, #252	0905
			10	12	0001B	BNEQ	1\$	
		04	AC	DD	0001D	PUSHL	ADDRESS	0909
			52	DD	00020	PUSHL	R2	
			52	DD	00022	PUSHL	R2	
		0000'	CF	9F	00024	PUSHAB	P.ABG	
	63		04	FB	00028	CALLS	#4, SYS\$FAO	
			17	11	0002B	BRB	2\$	
7E	04	AC	0A	00	EF	EXTZV	#0, #10, ADDRESS, -(SP)	0914
7E	05	AC	06	02	EF	EXTZV	#2, #6, ADDRESS+1, -(SP)	
			52	DD	00039	PUSHL	R2	
			52	DD	0003B	PUSHL	R2	
		0000'	CF	9F	0003D	PUSHAB	P.ABI	
	63		05	FB	00041	CALLS	#5, SYS\$FAO	0916
	50		62	9E	00044	MOVAB	DESC, R0	0918
			04	00047	2\$:	RET		

; Routine Size: 72 bytes, Routine Base: \$CODE\$ + 04AA



: 753 0919 1 END  
: 754 0920 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWN\$	52	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	696	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1266	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	15	0	581	00:01.0
\$255\$DUA28:[SHRLIB]NET.L32;1	1279	39	3	63	00:00.9

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SHONET/OBJ=OBJ\$:SHONET MSRC\$:SHONET/UPDATE=(ENH\$:SHONET)

: Size: 1266 code + 748 data bytes

: Run Time: 00:24.7

: Elapsed Time: 01:20.3

: Lines/CPU Min: 2237

: Lexemes/CPU-Min: 21745

: Memory Used: 183 pages

: Compilation Complete



0056

AH-BT13A-SE  
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY